CIS SOFTWARE WORKSHOP

TESTING YOUR SOFTWARE

DR. MUSTAFA BOZKURT

SESSION OBJECTIVES

- Understand why software testing is important
- Learn what a test case is
- Learn about a range of commonly used software testing techniques and strategies
- Learn about commonly used testing tools
- Basic Junit testing tutorial

SOFTWARE TESTING:

Software Testing is:

The execution of software for purposes of validation and verification. (ISTQB)

The process of executing a program or system with the intent of finding errors. (Myers)

Any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results (Hetzel)

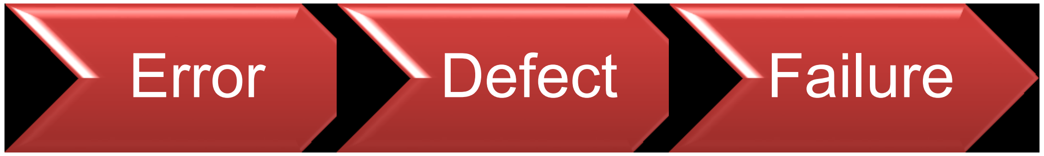
An investigation conducted to provide stakeholders with information about the quality of the product or service under test. (Kaner)

SOFTWARE TESTING: BUG, DEFECT, ERROR & FAILURE

A **failure** is an unacceptable behavior exhibited by a system.

A **defect** (a.k.a. **bug**) is a flaw in any aspect of the system including the requirements, the design and the code, that contributes, or may potentially contribute, to the occurrence of one or more failures.

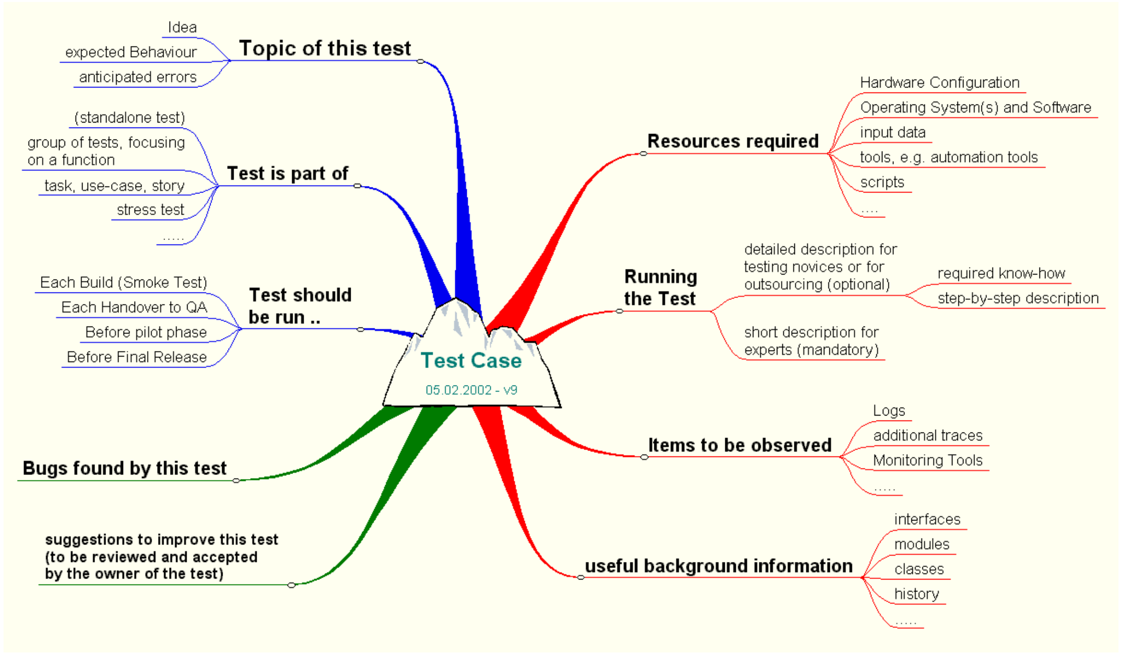
An **error** (a.k.a. **mistake**) is a slip-up or inappropriate decision by a software developer that leads to the introduction of a defect into the system



SOFTWARE TESTING: TEST CASE

Requires test data (inputs) and knowledge of the expected output.

A test case is the tuple (i, o),where i is the test data and o=S(i) for S the Software Under Test (SUT).



http://2.bp.blogspot.com/_vdqOsYKAf0Y/SxKr7eSIdbI/AAAAAAAAAp0/TshJWzz-5V0/s1600/TestCase_new.gif

TEST CASES EXAMPLE

Specifications of software to test: Capitalise a given word (no spaces allowed)

Input (i)	Expected Output (o)	Test Case
Spurs	SPURS	(Spurs, SPURS)
Tottenham Hotspur	error	(Tottenham Hotspur, error)
defoeJ	DEFOEJ	(defoeJ , DEFOEJ),
spurs3	SPURS3	(spurs3, SPURS3),
spurs Rule OK	error	(spurs Rule OK, error)

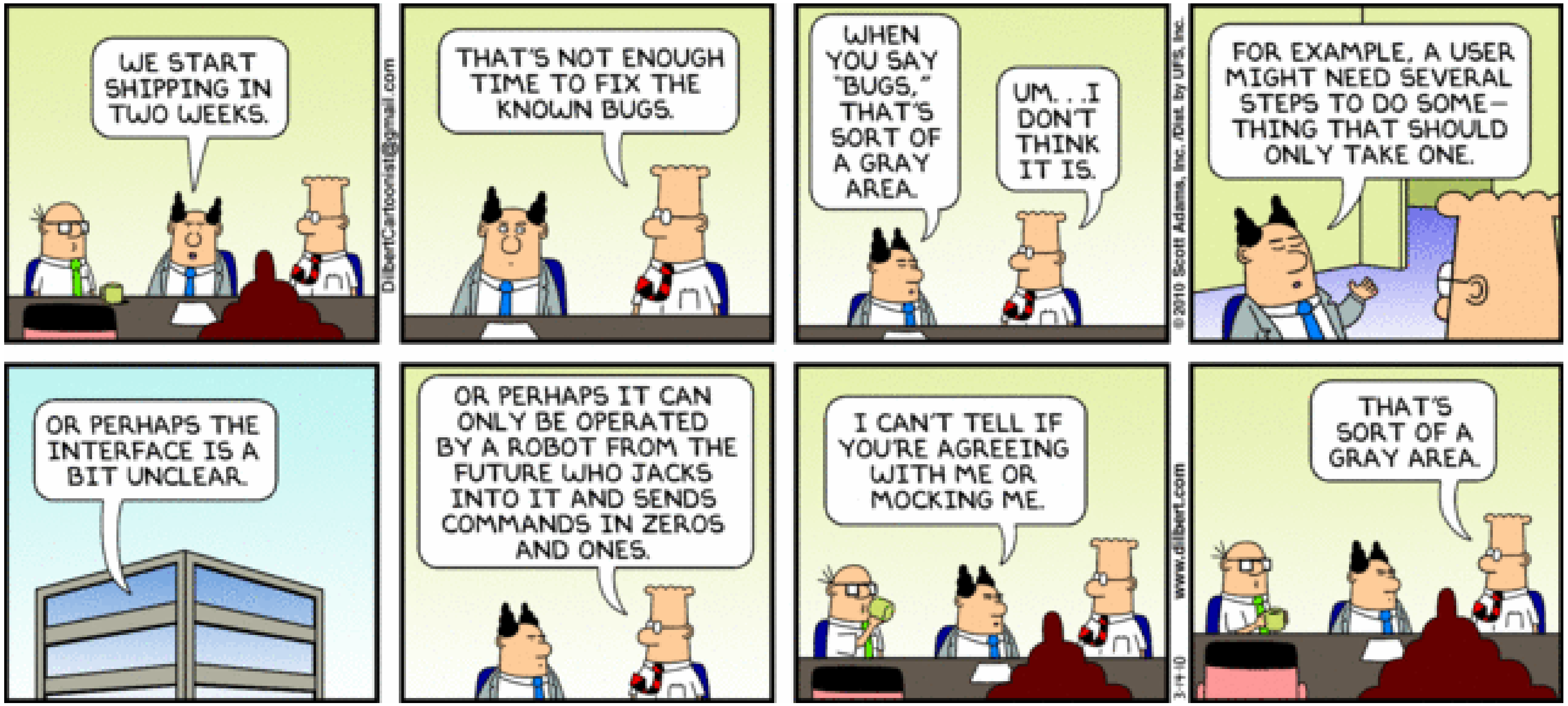
SOME REALLY BORING OBSERVATIONS

It is impossible to completely test any nontrivial module or any system.

- Halting problem
- Cost and effort

"Testing can only show the presence of bugs, not their absence" (Dijkstra)

- Except fault-based testing
  - Testing is fault-based when it seeks to demonstrate that prescribed faults are not in a program. (A Theory of Fault-Based Testing, L. J. Morell, IEEE Transactions on Software Engineering, 1990)


WE START SHIPPING IN TWO WEEKS.
THAT'S NOT ENOUGH TIME TO FIX THE KNOWN BUGS.
WHEN YOU SAY "BUGS," THAT'S SORT OF A GRAY AREA.
UM. . .I DON'T THINK IT IS.
FOR EXAMPLE, A USER MIGHT NEED SEVERAL STEPS TO DO SOME-THING THAT SHOULD ONLY TAKE ONE.
OR PERHAPS THE INTERFACE IS A BIT UNCLEAR.
OR PERHAPS IT CAN ONLY BE OPERATED BY A ROBOT FROM THE FUTURE WHO JACKS INTO IT AND SENDS COMMANDS IN ZEROS AND ONES.
I CAN'T TELL IF YOU'RE AGREEING WITH ME OR MOCKING ME.
THAT'S SORT OF A GRAY AREA.
DilbertCartoonist@gmail.com
© 2010 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com
3-14-10

MY CODE DOESN'T WORK
I HAVE NO IDEA WHY
MY CODE WORKS
I HAVE NO IDEA WHY
9GAG is your best source of fun.

TESTING
THE "STANDARD" APPROACH

1. Write some code
2. Get scared enough about some aspect of it to feel the need to 'try it out'
3. Write some kind of driver that invokes the bit of code. Add a few print statements to verify it's doing what you think it should
4. Run the test, eyeball the output and then delete (or comment out) the print statements
5. Go back to step 1

- Andy Hunt and Dave Thomas, IEEE Software 19(1), 2002

TESTING IN DEVELOPMENT

- Component testing
    - Unit or module testing
    - Integration testing
- Function testing
- System testing
- Regression testing

TESTING STRATEGIES

Bottom-up

- Test Driver needed (Unit testing is example)

Top-down

- Test stub needed

WHITE-BOX TESTING

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and **structural testing**) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing).

Wikipedia

Determining test cases from a knowledge of the internal logic of the software.

CODE COVERAGE

Code coverage analysis is the process of:

- Finding areas of a program not exercised by a set of test cases
- Creating additional test cases to increase coverage
- Determining a quantitative measure of code coverage, which is an indirect measure of quality

An optional aspect of code coverage analysis is:

- Identifying redundant test cases that do not increase coverage

WHITE-BOX TESTING TECHNIQUES

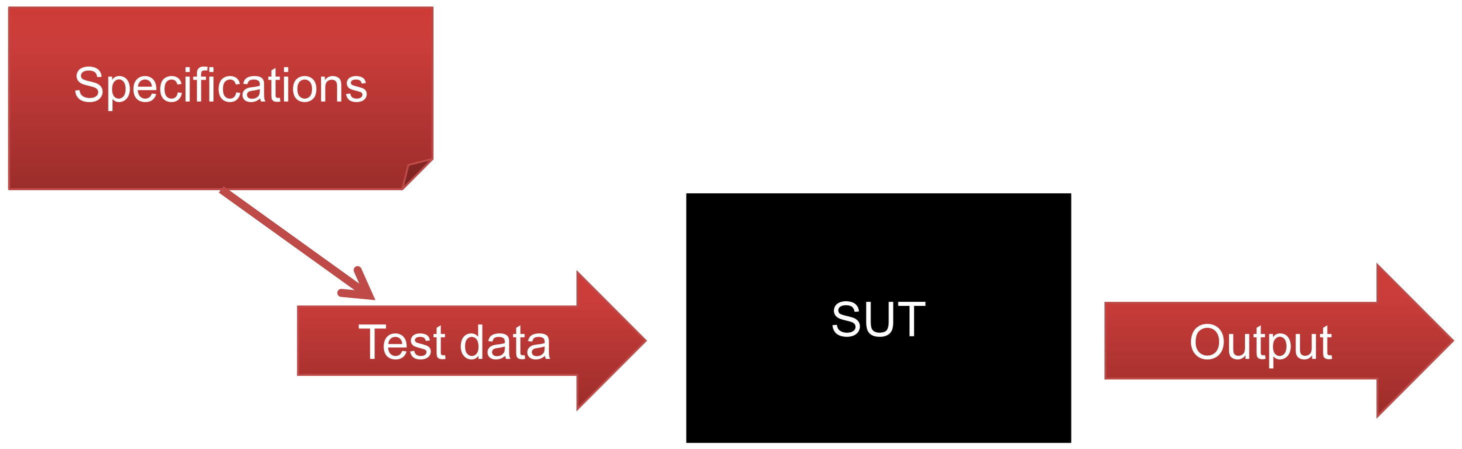
- Statement coverage
- Branch coverage
- Condition coverage
- Modified condition/decision coverage (MC/DC)
- Path coverage

BLACK-BOX TESTING

Black-box (also functional) testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings.

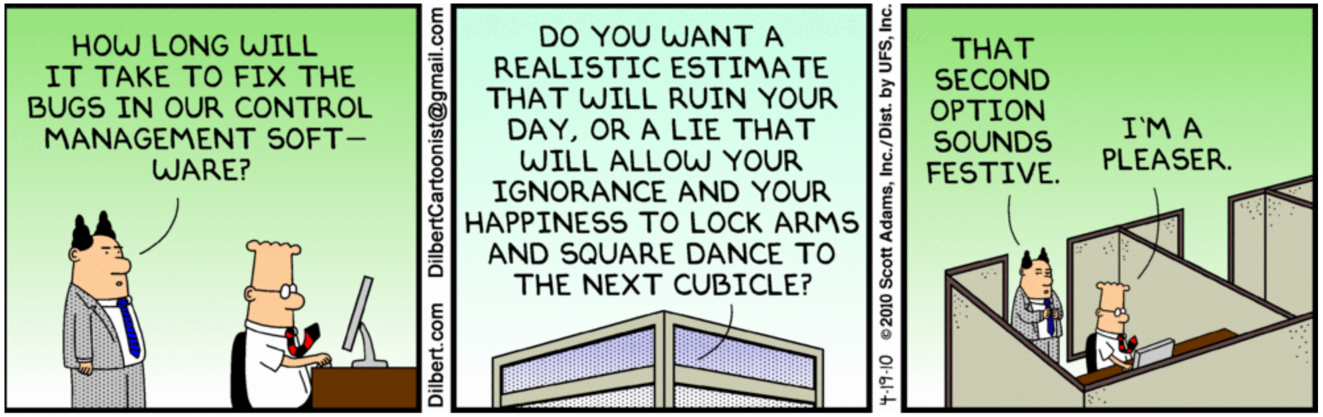
Wikipedia

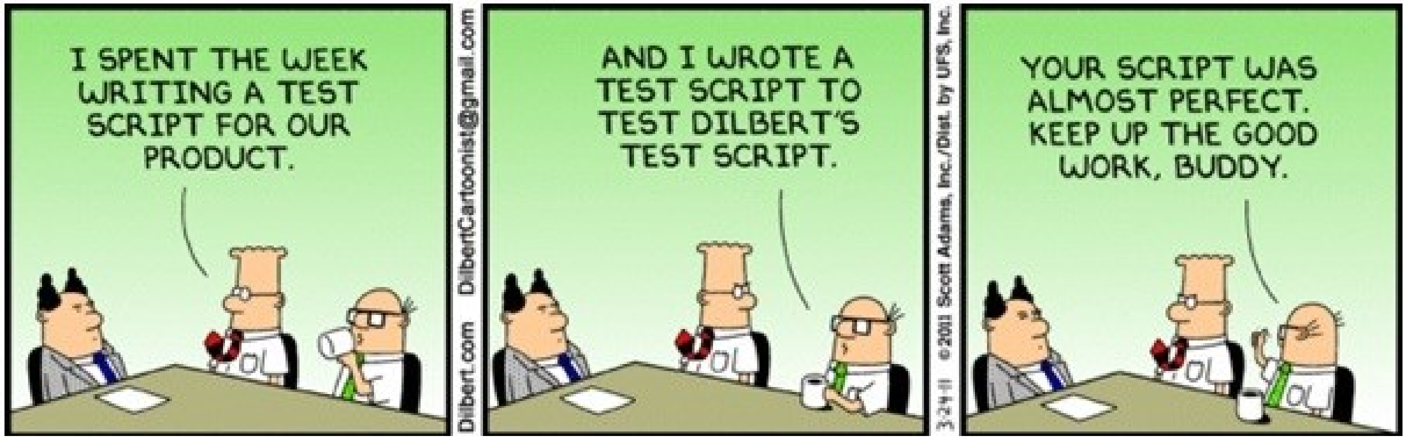
Determining test cases without the knowledge of the internal logic of the software



BLACK-BOX TESTING

- Equivalence partitioning
- Boundary value analysis
- Statistical testing
- Cause-effect graphing
- Error guessing
- Random testing
- Syntax directed testing
- State transition testing


HOW LONG WILL IT TAKE TO FIX THE BUGS IN OUR CONTROL MANAGEMENT SOFT-WARE?
DO YOU WANT A REALISTIC ESTIMATE THAT WILL RUIN YOUR DAY, OR A LIE THAT WILL ALLOW YOUR IGNORANCE AND YOUR HAPPINESS TO LOCK ARMS AND SQUARE DANCE TO THE NEXT CUBICLE?
THAT SECOND OPTION SOUNDS FESTIVE.
I'M A PLEASER.
Dilbert.com DilbertCartoonist@gmail.com
4-19-10 ©2010 Scott Adams, Inc./Dist. by UFS, Inc.


I SPENT THE WEEK WRITING A TEST SCRIPT FOR OUR PRODUCT.
AND I WROTE A TEST SCRIPT TO TEST DILBERT'S TEST SCRIPT.
YOUR SCRIPT WAS ALMOST PERFECT. KEEP UP THE GOOD WORK, BUDDY.
Dilbert.com DilbertCartoonist@gmail.com
3-24-11 ©2011 Scott Adams, Inc./Dist. by UFS, Inc.

CAN YOU TEST THE SOFTWARE TODAY?
NO, I'M MAKING MAJOR CHANGES TUESDAY.
YOU COULD TEST THE CURRENT VERSION.
I WISH PEOPLE WOULDN'T SLAP THEIR FOREHEADS AND SAY "AYE-YI-YI-YI" EVERY TIME I TALK.
www.dilbert.com scottadams@aol.com
© 2000 United Feature Syndicate, Inc.

TESTING AUTOMATION

Manual testing	Automated testing
Executing the test cases manually without any tool support is known as manual testing. Time consuming and tedious: Since test cases are executed by human resources so it is very slow and tedious. Huge investment in human resources: As test cases need to be executed manually so more testers are required in manual testing. Less reliable: Manual testing is less reliable as tests may not be performed with precision each time because of human errors. Non-programmable: No programming can be done to write sophisticated tests which fetch hidden information.	Taking tool support and executing the test cases by using automation tool is known as automation testing. Fast Automation runs test cases significantly faster than human resources. Less investment in human resources: Test cases are executed by using automation tool so less tester are required in automation testing. More reliable: Automation tests perform precisely same operation each time they are run. Programmable: Testers can program sophisticated tests to bring out hidden information.

UNIT TESTING FRAMEWORK

- **C++:** GoogleTest, CppUnit, Boost, CxxTest, Aeryn, Fructose
- **Java**: Junit, TestNG
- **Php**: PhPUnit
- **Python**: Unittest, pyTest

JUNIT FRAMEWORK

- JUnit is a unit testing framework for the Java programming language.
- JUnit is the most commonly used Java testing framework!

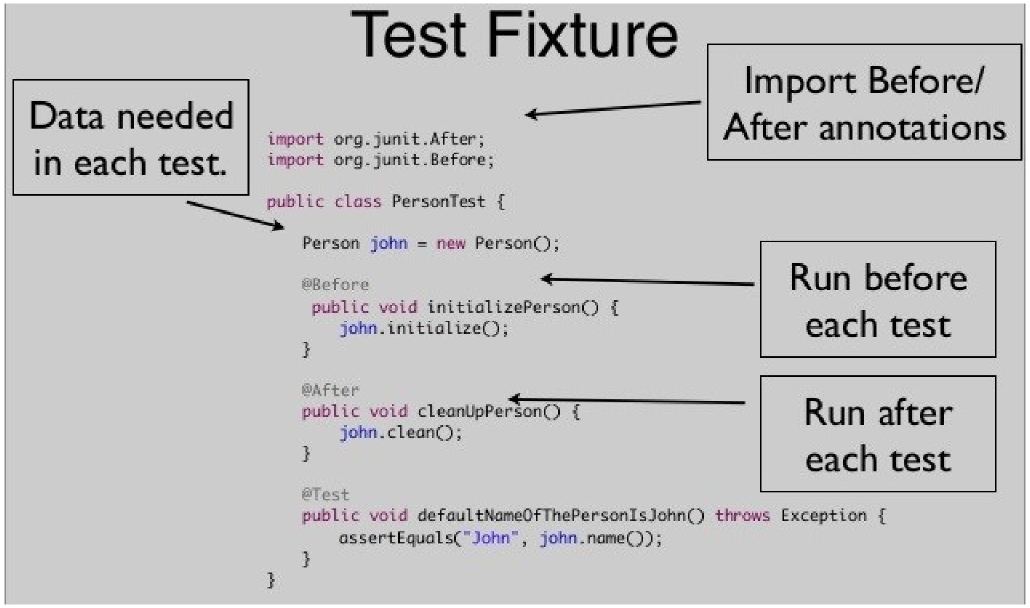
JUNIT FRAMEWORK

- Provides Annotation to identify the test methods.
- Provides Assertions for testing expected results.
- Provides Test runners for running tests.
- JUnit tests allow you to write code faster which increasing quality
- JUnit is elegantly simple. It is less complex & takes less time.
- JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results.
- JUnit tests can be organized into test suites containing test cases and even other test suites.
- Junit shows test progress in a bar that is green if test is going fine and it turns red when a test fails.

JUNIT TERMINOLOGY

A **test fixture** sets up the data (both objects and primitives) that are needed to run tests

Example: If you are testing code that updates an employee record, you need an employee record to test it on



```
import org.junit.After;
import org.junit.Before;

public class PersonTest {

    Person john = new Person();

    @Before
     public void initializePerson() {
        john.initialize();
    }

    @After
    public void cleanUpPerson() {
        john.clean();
    }

    @Test
    public void defaultNameOfThePersonIsJohn() throws Exception {
        assertEquals("John", john.name());
    }
}
```

JUNIT TERMINOLOGY

A **unit test** is a test of a single class

A **test case** tests the response of a single method to a particular set of inputs

A **test suite** is a collection of test cases

A **test runner** is software that runs tests and reports results

JUnit provides some limited support for integration tests!